# Curso básico de Programação em Python

## Encontro 8 - Manipulação de arquivos

Prof. Louis Augusto
louis.augusto@ifsc.edu.br


Instituto Federal de Santa Catarina
Campus São José

# Índice

# Sumário

# Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

Arquivos de texto: são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

Arquivos binários: são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

# Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

**Arquivos de texto:** são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

**Arquivos binários:** são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

# Arquivos de texto e arquivos binários

Há três tipos de arquivos que podem ser lidos para obtenção de dados, arquivos de texto e binários.

Arquivos de texto: são arquivos que podem ser lidos por pessoas, como este texto, e são editáveis por editores de texto, como o Pluma ou Writer (no Linux) ou Bloco de Notas ou Word (no Windows). O preço a ser pago é que a leitura pelo computador é lenta. Normalmente tem extensões .txt, .dat, .doc etc.

Arquivos binários: são arquivos que não podem ser lido por pessoas, precisam ser lidos, e possivelmente editados, por um programa executável, sendo que este deve conhecer muito bem o formato em detalhes. São rápidos para salvar e carregar em comparação com arquivos de texto. Como exemplos temos arquivos de imagem .jpg, de vídeo .mp4, de áudio .mp3 etc.

# Sumário

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

## Operação (A função open() )

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

**Operação (A função open() )**

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

## Operação (A função open() )

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

## Operação (A função open() )

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

## Operação (A função open() )

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Até o momento gravamos ou lemos informações usando a forma padrão de entrada e saída do Python (input() e print()) e redirecionamento.

A partir de agora será conveniente guardar informações de forma mais especializada. A linguagem Python oferece funções que servem para manipular arquivos sem muitas dificuldades.

## Operação (A função open() )

*A sintaxe básica da função built-in open() é:*

```
Objeto_arquivo = open(nome_do_arquivo, modo, buffer )
```

Um exemplo básico de abertura é:

```
#-*- coding: utf-8 -*-
# Abertura de arquivo
fo = open("foo.txt", "wb")
```

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Abertura de arquivos

Temos os elementos:

- `Objeto_arquivo`: Nome da variável que referencia o arquivo. A partir da leitura o arquivo é referenciável não por ser nome, mas pela variável.
- `Nome_do_arquivo`: é a string que recebe o nome do arquivo em disco.
- `modo`: é o tipo de arquivo que será manipulado, de texto ou binário, e se será um arquivo para leitura, escrita (podendo ser criado ou alterado). O próximo slide trabalhará este item.
- `buffer`: serve para indicar se a manipulação do arquivo será feita diretamente pelo sistema operacional ou não, podendo implicar numa leitura mais rápida. É um tópico avançado, que não será trabalhado aqui. Pode-se deixar o campo vazio ou usar -1 para o valor default.

Quando abrimos um arquivo recebemos um cursor, (ponteiro de posicionamento de arquivo), que indica a posição em que se está no arquivo, como um cursor num editor de texto.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *r* Abre um arquivo texto para somente leitura. O ponteiro de posição para o arquivo (como se fosse um cursor num editor de texto) se localiza no início do arquivo.
- *rb* Abre um arquivo binário para somente leitura. O ponteiro para o arquivo se localiza no início do arquivo.
- *rb+* Abre um arquivo binário para leitura e escrita. O ponteiro para o arquivo se localiza no início do arquivo.
- *w* Abre um arquivo texto para escrita. Se já houver um arquivo com o mesmo nome será sobrescrito.
- *wb* Abre um arquivo para somente escrita em format binário. Sobrescreve o arquivo caso já exista.
- *w+* Abre um arquivo em modo texto para leitura e escrita, sobrescreve o arquivo se já existir.
- *wb+* Abre um arquivo em modo binário para leitura e escrita em formato binário, sobrescreve o arquivo se já existir.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo (*a* = *append*, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

# Modo de manipulação de arquivos

Há vários modos para manipulação de arquivos:

- *a* Abre um arquivo para anexação de informação, ou seja, permite inclusão de informação no final do arquivo ($a = append$, que siginifica anexar). O cursor é posicionado no final do arquivo para anexação. Caso o arquivo não exista é criado um arquivo novo para escrita.
- *a+* Abre um arquivo para leitura e anexação. O ponteiro de posição se inicia no fim do arquivo, se existir, caso contrário cria um arquivo novo para escrita.
- *ab* Abre um arquivo para anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, caso contrário cria um arquivo para escrita.
- *ab+* Abre um arquivo para leitura e anexação em modo binário. O cursor é posicionado no final do arquivo, caso exista, do contrário cria um novo arquivo para leitura e escrita.

# Sumário

# Exemplos

Vamos criar um arquivo texto bem simples.

```
#-*- coding: utf-8 -*-
t_str= "Eu gosto de Python. "
t2_str = "Vamos estudar python. "

#abertura e gravacao
ft=open("ArqTexto.dat",'w')
ft.write(t_str)
ft.write('\n')
ft.write(t_str)
ft.write('\n')
ft.close()
```

Abra o arquivo e veja como ficou.

# Exemplos

Vamos criar um arquivo texto bem simples.

```
#-*- coding: utf-8 -*-
t_str= "Eu gosto de Python. "
t2_str = "Vamos estudar python. "

#abertura e gravacao
ft=open("ArqTexto.dat",'w')
ft.write(t_str)
ft.write('\n')
ft.write(t_str)
ft.write('\n')
ft.close()
```

Abra o arquivo e veja como ficou.

# Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

# Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

# Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

# Exemplo em modo texto

Faça uma anexação ao arquivo:

```
ft2 = open("ArqTexto.dat",'a')
ft2.write(t2_str)
ft2.write('\n')
ft2.close()
```

Abra novamente o arquivo e veja como ficou.

Vamos agora abrir o arquivo em modo texto:

```
#Abertura e leitura
ft2 = open("ArqTexto.dat",'r')
linhas = ft2.readlines()
ft2.close()
#Impressão
for i in range(len(linhas)):
print(linhas[i],end = '')
print()
```

# Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

# Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

## Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

# Exemplo em modo binário

```
#Modo binário:
t_strb= "Eu gosto de Python em modo binario. \n"
t2_strb = "Vamos estudar python em modo binario. "
primos = [2, 3, 5, 7]
#Conversao do array para binario
b_str= bytearray(t_strb, 'utf-8')
b2_str= bytearray(t2_strb,'utf-8')
#Funcao bytearray converte o arquivo para binario,
tem-se que indicar a codificacao.
```

## Gravação

```
fb1 = open("ArqBin.b2n",'wb')
fb1.write(b_str)
fb1.close()
```

## Anexação:

```
fb2 = open("ArqBin.b2n",'ab')
fb2.write(b2_str)
fb2.close()
```

# Exemplo em modo binário

```
b3 = open("ArqBin.b2n",'rb')
lines = fb3.readlines() #Leitura
fb3.close()

for i in range(len(lines)):
print(lines[i],end = '')
print()

#Para converter em strings usamos a função decode()
for i in range(len(lines)):
print(lines[i].decode(),end = '')
print()
```